

Van testen naar ontwerpen

Test- en herstelkosten terugdringen

Softwarebouwers weten wel dat ontwerpen en inspecties belangrijk zijn, maar in de praktijk komt er vaak weinig van terecht. Benchmarkstudies laten zien dat de kosten daardoor veel hoger uitvallen. Een herverdeling van projectinspanningen is daarom gewenst.

Hans Sassenburg en Peter Wijnhoven

Na een jaar waarin de software-industrie rake klappen kreeg te verwerken, trekt de markt weer aan. Maar hebben we ook wat geleerd van deze crisis? Of zijn we net zo argeloos als de financiële industrie, waar per saldo niets veranderd lijkt te zijn?

In de jaren zeventig van de vorige eeuw introduceerden David Parnas en Michael Fagan belangrijke principes en technieken. Veel mensen zijn bekend met hun werk, maar de toepassing ervan laat te wensen over. Sterker nog, ze lijken steeds minder toegepast te worden. Hoe komt dit?

En belangrijker nog: wat kan hieraan worden gedaan?

Belang van ontwerp en inspecties

Parnas zag dat het ontwerp een steeds belangrijkere rol ging spelen bij de grotere en complexere softwaresystemen. Hij introduceerde daarom drie ontwerpprincipes: high cohesion, loose coupling en information hiding. Hij is groot voorstander van het in kaart brengen van moeilijke ontwerpbesluiten. Met behulp van deze principes wordt vervolgens naar een oplossing gezocht, waarbij modules zo onafhankelijk mogelijk van elkaar worden gedefinieerd. Deze principes zijn nog steeds uiterst actueel en hebben twee belangrijke voordelen: ze dragen bij aan een ontwerp dat werkelijk realiseerbaar is en ze maken een ontwerp robuust voor toekomstige aanpassingen.

Michael Fagan publiceerde in 1976 een artikel over het belang van inspecties. Hij analyseerde het ontwikkelproces van software en stelde vast dat latere projectfasen zoals testen niet erg effectief zijn in het opsporen van fouten. Bovendien is het verhelpen van dan gevonden fouten uiterst kostbaar. Hij introduceerde inspecties die als doel hebben in vroege projectfasen fouten aan het licht te brengen. Inspecties kunnen worden toegepast op specificaties, ontwerp en code.

Cost-of-Quality-model

De principes van Parnas en Fagan zijn redelijk bekend, maar worden ze ook toegepast? Om deze vraag te kunnen beantwoorden kan het Cost-of-Quality-model worden gebruikt. Dit model is gebaseerd op ideeën van de kwaliteitsgoeroes Joseph Juran en Philip Crosby. Een door de auteurs gemaakte variant voor softwareontwikkeling maakt onderscheid in vier kostencategorieën: 'Basis', 'Ondersteuning', 'Preventie' en 'Test/herstel' (zie kader). De ontwerpactiviteiten van Parnas vallen in de categorie 'Basis', ze zijn immers essentieel en dragen bij aan de directe waarde van het product voor de klant. De inspectieactiviteiten vallen in de categorie 'Preventie', ze zijn gericht op het voorkomen van slechte kwaliteit.

Samenvatting

Om te zien of de ontwerp- en inspectieprincipes van Parnas en Fagan worden toegepast, kan gebruik worden gemaakt van het Cost-of-Quality-model. In een variant voor softwareontwikkeling worden de categorieën 'Basis', 'Ondersteuning', 'Preventie' en 'Test/herstel' onderscheiden. Een hoge test- en herstelinspanning is te voorkomen door bij het plannen van een project de gewenste verhouding tussen de vier categorieën vast te stellen.

Een interessante vraag is wat een goede verhouding is van een projectbudget over deze vier componenten. Een nog interessantere vraag is echter wat de verhoudingen in de praktijk zijn. Hiertoe kan een benchmarkstudie worden uitgevoerd (zie kader). Het grote voordeel van een benchmarkstudie is dat een organisatie met harde getallen wordt geconfronteerd: de eigen manier van werken wordt afgezet tegen industriewaarden. Men weet wel dat een goed ontwerp en inspecties belangrijk zijn, maar in de waan van de dag wordt reactief gereageerd op operationele problemen. De uitkomst van zo'n studie leert wat het effect van deze veronachtzaming is. Te weinig tijd besteden aan ontwerp en inspecties leidt tot factoren hogere kosten tijdens latere projectfasen. Op basis van zo'n uitkomst kan vervolgens worden gekeken wat zinvolle verhou-

dingen voor de betreffende organisatie zijn. Deze verhoudingen kunnen bij aanvang van een project geverifieerd worden en tijdens de projectuitvoering bewaakt. Op deze wijze wordt op basis van getallen afgedwongen dat tijdens een project de tijd aan de juiste activiteiten wordt besteed.

»Te weinig tijd besteden aan ontwerp en inspecties leidt tot veel hogere kosten tijdens latere projectfasen«

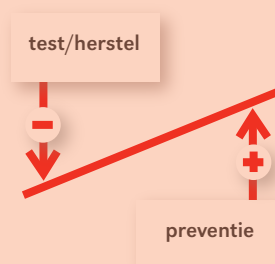
Cost-of-Quality-model: preventie loont

Projectkosten kunnen onderverdeeld worden in vier categorieën:

- **Basis:** kosten die direct bijdragen aan de waarde van het product voor de klant (specificeren, ontwerpen, implementeren).
- **Ondersteuning:** kosten die indirect bijdragen aan de waarde van het product voor de klant (projectmanagement, configuration management, administratieve ondersteuning).
- **Preventie:** kosten als gevolg van het voorkomen van slechte kwaliteit (reviews, inspecties, opleiding en training).
- **Test/herstel:** kosten als gevolg van het vaststellen of een product aan kwaliteitseisen voldoet (testen, audits) en kosten veroorzaakt door het verwijderen van fouten (herstelwerkzaamheden, afhandelen van klachten van klanten, aansprakelijkheidsclaims, foutreparaties, terughalen van producten).

Basis- en ondersteuningskosten zijn essentiële kosten, in die zin dat ze niet te vermijden zijn. Preventie- en test- en herstelkosten zijn niet-essentiële kosten gericht op het behalen van een zekere kwaliteit. Het is de uitdaging de totale niet-essentiële kosten zo laag mogelijk te houden. Investeren in preventieve activiteiten leidt

in het algemeen tot een aanzienlijke vermindering van test- en herstelkosten (hefboomwerking, zie figuur 1).

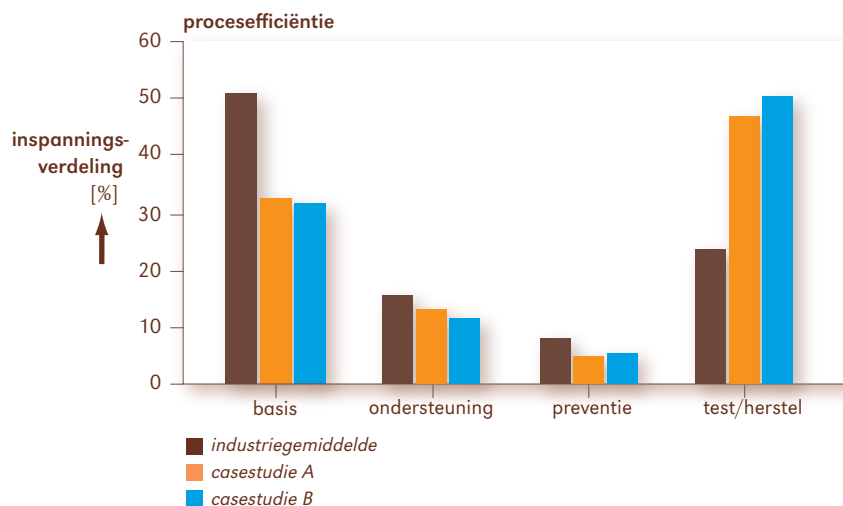


Figuur 1. Hefboomwerking tussen investeringen in preventie en test- en herstelkosten

Benchmarkstudies: Cost-of-Quality-indicatoren verfijnen inzicht

In benchmarkstudies worden de waarden van verschillende key performance indicators (KPI's) in kaart gebracht. Als referentie worden een of meer projecten geselecteerd. Daarnaast worden de gevonden waarden afgezet tegen gemiddelde industriewaarden. Deze zijn verschillend per industrie-segment. Vaak wordt een benchmarkstudie beperkt tot drie KPI's: doorlooptijd, kosten en gevonden fouten na productvrijgave. Het is echter zinvol deze set uit te breiden met minimaal vier KPI's, namelijk de waarden van de vier categorieën uit het Cost-of-Quality-model. Dit is niets anders dan de uitsplitsing van de totale ontwikkelkosten naar deze categorieën. Figuur 2 geeft een voorbeeld van de uitkomst van een benchmarkstudie voor twee projecten (casestudie A en B). Beide casestudies

betroffen embedded softwareontwikkeling van middelgrote systemen. Hierbij is te zien dat gemiddeld 50 procent van het projectbudget gespendeerd werd aan test- en herstelwerkzaamheden. Dit is een uiterst ongezonde verhouding en duidt op een zéér inefficiënt ontwikkelproces. In deze 50 procent zijn bovendien alleen meegenomen de test- en herstelkosten vóór productvrijgave. Het meenemen van de afhandeling van klachten van klanten na productvrijgave zou tot een verdere verhoging leiden. De gevonden waarden zijn afgezet tegen gemiddelde industriewaarden. Hierbij is te zien dat 25 procent voor test- en herstelwerkzaamheden normaal is. Dit gemiddelde is berekend uit de data van veel bedrijven, groen en rijp door elkaar.



Figuur 2. Uitkomst van een benchmarkstudie gebaseerd op Cost-of-Quality-model

Concrete activiteiten

Terug naar Parnas en Fagan. Ervan uitgaande dat in een projectplan voldoende ruimte is ingebouwd voor ontwerp en inspecties, welke concrete activiteiten kunnen dan worden onderscheiden? Het boek *Evaluating Software Architectures, Methods and Case Studies* (Clements e.a., 2002) biedt goede richtlijnen:

- Active Design Reviews (ADR) / Active Reviews for Intermediate Designs (ARID). ADR is oorspronkelijk opgezet door Parnas. Het Software Engineering Institute (SEI) heeft deze aanpak later verfijnd in de zogenaamde ARID-aanpak. Het is een methode om nog niet gedocumenteerde ontwerpen van deelsystemen in ontwikkeling te reviewen. De kern van beide aanpakken is om

actieve deelname af te dwingen en het review-proces te structureren. Dit wordt bereikt door gericht reviewers te identificeren op basis van hun expertise en interactie met het systeem. Vervolgens bestuderen en analyseren deze reviewers met specifiek opgestelde checklists en/of scenario's de subarchitectuur in wording.

- **Architecture Tradeoff Analysis Method (ATAM).** ATAM is door het SEI ontwikkeld en is een methode om architectuurvoorstellen te evalueren. Hierbij wordt gekeken in welke mate een voorgestelde architectuur voldoet aan kwaliteitseisen. Voorbeelden hiervan zijn uitbreidbaarheid, overdraagbaarheid en onderhoudbaarheid. Ook ATAM is een goed gedefinieerde aanpak met een vastgelegd proces voor een geselecteerde groep deelnemers. Het kan gezien worden als een Active Design Review op architectuurniveau.
- **Inspecties.** Deze door Fagan geïntroduceerde reviewaanpak wordt meestal gebruikt voor documenten (ontwerpen) en code. Hij is echter ook uitstekend geschikt om in kleine teams het detailontwerp of de code van delen uit het systeem te inspecteren en fouten op te sporen.

Zoals gezegd zijn deze aanpakken niet nieuw. Het lijkt er zelfs op dat ze in de praktijk steeds minder worden toegepast. De meeste projecten beperken zich tot eenmalige reviews van slechts enkele documenten. Uiteindelijk wordt men dan onvermijdelijk geconfronteerd met een hoge test- en herstelinspanning. Dit kan voorkomen worden door bij het plannen van een project de gewenste verhouding tussen de vier geschetste categorieën vast te stellen. Hiertoe dient bij voorkeur historische informatie uit eigen projecten als referentie te worden genomen. Vervolgens kan men concrete verbeteringen definiëren en het effect daarvan op de categorieverhoudingen vaststellen. Hiernaast kunnen benchmarkgegevens gebruikt worden om het plan te vergelijken met industriewaarden. Tijdens het project is het daarna zaak de actuele verhoudingen in relatie tot het plan te bewaken. Blijkt dat de actuele inspanning voor preventies achterblijft bij de geplande waarde, dan is dit een signaal voor het management. Zonder tegenmaatregelen is de kans op een hoge testinspanning erg groot.

Op deze wijze wordt door middel van reeds langer bestaande aanpakken een 'lean' ontwikkelmodel ingevuld. Er wordt continu gekeken waar mogelijk een bron van verspilling (te veel testen, herstelwerkzaamheden) ontstaat. En de hier besproken technieken zijn een hefboom om deze verspilling tegen te gaan. Het Cost-of-Quality-model is hierbij het kwantitatieve hulpmiddel.

Literatuur

Clements, P. e.a. (2002). *Evaluating Software Architectures, Methods and Case Studies*. Addison-Wesley.

Dr. ir. Hans Sassenburg

is zelfstandig adviseur en is als visiting scientist verbonden aan het Software Engineering Institute. Hij richtte in 2009 de Software Benchmarking Organization (SBO) op. E-mail: hans@se-cure.ch.

Ing. Peter Wijnhoven

is managing consultant bij Sioux Embedded Systems BV. Sioux is officieel partner van SBO in Nederland. E-mail: peter.wijnhoven@sioux.eu.